采用类心密度策略的多目标微分自动聚类算法

申晓宁，孙　毅，薛云勇，孙　帅

(南京信息工程大学 信息与控制学院，南京 210044)

摘　要：针对聚类过程中，由于类心选取的随机性导致所选类心偏离数据集，或者类心过于集中而带来的错误聚类这一缺陷的研究，所提算法对类心的选取进行两次筛选，即将类心密度过小的以及两两类心之间距离过小的类心分别筛选出来，不让其参与聚类，此后算法对筛选后剩余的类心再进行聚类。为了使算法能较快地得到最优类心，提出了改进的聚类准则函数，对聚类数目进行动态地惩罚。为了评估所提算法在聚类问题上的应用性能，选择两种不同类型的数据集进行了仿真实验。与其他三种现有的自动聚类算法的比较结果表明，所提算法能够获得更好的聚类结果，从而验证了算法所提策略的有效性。

关键词：自动聚类；类心密度策略；类心筛选；多目标优化；微分进化

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2018.04.0288

Multi-objective differential evolution automatic clustering algorithm based on class-center density

Shen Xiaoning, Sun Yi, Xue Yunyong, Sun Shuai

(*School of Information& Control*, *Nanjing University of Information Science & Technology*, *Nanjing* 210044, *China*)

Abstract: In the process of clustering, for the reason that the randomness of the class-center selection may lead to the phenomenon that the selected class-center deviates from the data set, or the class-center is too centralized, the proposed algorithm selects the class-center for two times: The class-centers which have too small density or have small distances between pairs of class-centers are screened out, and the algorithm does not allow them to participate in clustering. Then the algorithm continues to cluster the remaining class-centers. In order to make the algorithm get the optimal class-center quickly, we propose an improved clustering criterion function to penalize the number of clusters dynamically. In order to evaluate the performance of the proposed algorithm on clustering problems, experiments on two types of data sets are carried out. Compared with the other three existing automatic clustering algorithms, simulation experiments show that the proposed algorithm can obtain better clustering results, which validates the effectiveness of the proposed strategies.

Key words: automatic clustering; class-center density strategy; class-center screening; multi-objective optimization; differential evolution

0　引言

在科学研究和工程设计过程中，很多具体问题都可以归纳为参数优化问题。而现实当中，这些优化问题往往会有多个设计目标，这些目标互相矛盾，彼此制约。表现为一个目标的性能优化往往会使得其它至少一个目标的性能退化，即多个目标很难同时达到最优，通常称这类问题为多目标优化问题[1]。

微分进化算法（differential evolution，DE）是由 Store 和 Price 于 1997 年提出的一种基于群体差异的启发式并行搜索方法[2]。DE 算法由初始化、变异、交叉、选择等操作组成，区别于其他优化算法，DE 算法的进化个体扰动是通过多个个体的差分信息来体现的。DE 具有收敛速度快、控制参数少，并且结构简单、优化结果稳定等优点，因此，越来越多研究者开始将其应用到多目标优化问题的求解中

聚类分析技术作为一门数据分析工具和方法，在许多应用和研究领域都有着广阔的应用。然而现实中的大多数数据都是没有任何先验知识的，数据的类别数无法预先确定，这些问题可以归结为自动聚类问题[3]，即在不预先指定聚类个数的情况下对数据进行自动正确的聚类。

近年来，国内外学者提出了许多自动聚类算法。Das 等人在 2008 年提出基于改进微分进化的自动聚类算法 ACDE[4]，该算法对基本的微分进化算法中变异因子和交叉因子进行了改进，并且采用实数、定长的染色体编码方式，使得算法能够实现自动聚类。Maulik 等人在 2009 年和 2010 年分别提出了 ACDE 的

收稿日期：2018-04-20；**修回日期**：2018-06-20　　**基金项目**：国家自然科学基金资助项目（61502239）；江苏省自然科学基金资助

作者简介：孙毅（1991-），男，江苏淮安人，硕士研究生，主要研究方向为演化计算与应用研究；薛云勇（1993-），男，江苏泰兴人，硕士研究生，主要研究方向为大规模数据研究；孙帅（1992-），男，江苏泗洪人，硕士研究生，主要研究方向为机器人研究.

两个改进算法 ADEFC[5]和 MoDEAFC[6]，其中算法 ADEFC 加入了模糊划分测度，用模糊 C-均值聚类（FCM）来更新聚类中心；算法 MoDEAFC 在算法 ADEFC 基础上改进了微分进化的变异操作。上述算法的不足之处是仅用一个指标作为聚类准则函数，针对不同的数据集会有不同的聚类效果，算法的鲁棒性比较差。2014 年，Rodriguez 等人在《Science》上提出了密度峰聚类算法（RLCLu）[7]。密度峰聚类是一种新的基于密度的聚类算法， 该算法主要分为两个步骤：通过“决策图”人工选取密度峰，也即类心；分配剩余数据点得到聚类结果。该算法能够发现非球形簇，但却无法自动确定类心，特别是针对一些特殊数据集，通过决策图人工选取类心时容易出错。针对 RLCLu 算法这一缺陷，李涛提出了一种自动确定类心的密度峰聚类算法（ADPC）[8]。该算法主要分三步实现：a)计算每个数据点的局部密度和该点到具有更高密度数据点的最短距离；b)根据排序图自动确定类心；c)将剩下的每个数据点，分配到比其密度更高且距其最近的数据点所属的类别，并根据边界密度识别噪声点，得到聚类结果。然而，该算法的不足之处在于，面对一些簇内没有密度峰或同时具有多个密度峰的复杂流形结构数据集时，会出现能够自动确定类心却无法得到理想聚类结果的问题。为了解决算法 RLCLu 的缺陷，Ye 等人[9]亦对其进行了改进，主要针对 RLCLu 采用的“若一个点的距离偏差乘上该点的局部密度所得到的值最大，则选择该点为聚类中心”这一测度进行了改进，提出了进一步计算该点所得测度到集合中其它剩余点测度之间的绝对差值，从而扩大了聚类中心同其它点之间的差异性，为机器自动聚类奠定了基础，其不足是由于类心选择的随机性，也会出现能够自动确定类心却无法得到理想聚类结果的问题。

针对聚类过程中，由于类心选取的随机性易导致错误聚类这一缺陷，以及在事先不知道聚类个数的情况下，如何对数据进行自动而准确的聚类这些问题，本文在多目标微分进化算法的基础上，结合改进的类心密度策略以及聚类有效性指标，提出了采用类心密度策略的多目标微分自动聚类算法（A Multi-Objective Differential Evolution Automatic Clustering Algorithm Based on the Class-Center Density，MODEAC-CD）。MODEAC-CD 首先对类心进行两次筛选，即将类心密度过小的以及两两类心之间距离过小的类心分别筛选出来，不让其参与聚类；然后对剩下的类心，按照数据集中各数据距离某一类心最近的原则进行聚类。将该策略与多目标微分进化算法相结合，采用类内距离与改进的类间距离作为评价聚类质量的两个准则函数，有效地增加了聚类的准确性和收敛速度。

1　类心密度策略

1.1　个体编码方式

与传统的基于进化算法的聚类算法编码方式不同，本文采用一种基于实数、定长的染色体编码方式[4]。假设对具有 n 个点的数据集，每个数据点有 d 维，$K_{max}=\sqrt{n}$ 为每个体可能包含的最大聚类数[4]，则个体可以表示为：

$$\boldsymbol{V}=(T_1,\cdots,T_i,\cdots,T_{K_{max}},\boldsymbol{C_1},\cdots,\boldsymbol{C_i},\cdots,\boldsymbol{C_{K_{max}}}) \quad (1)$$

其中，类心 $\boldsymbol{C_i}=(C_{i1},C_{i2},\cdots,C_{id})$，$T_i$ 表示标签位阈值，与 $\boldsymbol{C_i}$ 一一对应，它决定了类心 $\boldsymbol{C_i}$ 是否被激活参与聚类。T_i 是实数，且 $T_i\in[0,1]$。

在该个体编码方式中，聚类中心的激活遵循一定的规则：当 $T_i>0.5$ 时，其所对应的类心 $\boldsymbol{C_i}$ 被激活，然后算法根据激活的类心进行聚类。群体中每个个体都由标签位阈值和类心两部分组成，因此，个体的总长度为 $K_{max}+K_{max}\times d$。前面的 K_{max} 个 T_i 表示标签位阈值，后面的 $K_{max}\times d$ 个基因位表示类心。

1.2　类心偏离数据集的改进

文献[7]介绍了如果一个数据点被看做是聚类中心，应该满足两个基本条件：该数据点被密度相对较低的邻域点所包围；该点与其他密度更高的点之间距离应相对较大。基于该思想，本节提出了针对类心偏离数据集的改进策略，并在 1.3 节给出了针对类心过于集中的改进策略，用以选择较好的聚类中心。

设进化算法的群体规模为 N，如前述，每个个体 $\boldsymbol{V}$ 含有 K_{max}个类心，因此一共有$(N)\times K_{max}$个类心（每个类心的每一维取值，均是在待聚类数据的每一维特征取值范围内随机生成）。针对类心偏离数据集的改进策略实现步骤如下：

a)将 $N\times K_{max}$ 个类心看做是待聚类的数据，并且计算两两类心之间的欧氏距离，得到一个距离矩阵 $M_{(N\times K_{max})\times(N\times K_{max})}$；

b)由距离矩阵 M 分别计算每一个类心与其他所有类心之间距离的均值，得到一个平均距离组记为$\{\boldsymbol{R_q}\}$，$q=1,2,\cdots(N)\times K_{max}$，该平均距离组$\{\boldsymbol{R_q}\}$能够很好地反映待聚类数据集中，各类心之间的分布情况。通常，$\{\boldsymbol{R_q}\}$中某一标量均值越小，则其所对应的类心处于数据集稠密区域的可能性越大；

c)计算$\{\boldsymbol{R_q}\}$的均值，得到一个标量均值记为R_2,根据一组数据的均值，能够反映出该组数据的集中程度这一特征，将 R_2看做是在待聚类数据集中统计所有类心拥有近邻个数的阈值；

d)对于第 q 个类心 $(q=1,2,\cdots(N)\times K_{max})$,找出与它之间的距离小于阈值$R_2$的其他类心个数，用数组 $\{\boldsymbol{D_q}\}$ 记录，$q=1,2,\cdots(N)\times K_{max}$，并将其看做是第 q 个类心的密度；

e)计算这个数组$\{\boldsymbol{D_q}\}$的均值，得到一个标量均值记为 R_5。从数组$\{\boldsymbol{D_q}\}$中找出类心密度小于阈值 R_5 的类心，并依次将与这些类心配对的标签位阈值修改为 0~0.5 的一个实数，目的是不让它们参与聚类。同样，将类心密度大于阈值 R_5的类心标签位阈值，依次修改为 0.5~1 的一个实数，目的是让它们参与聚类。

图 1 给出了所提算法 MODEAC-CD 在模拟数据集 long1[10]上筛选类心的过程，其中 x 轴，y 轴为各类心在 2 维空间上的坐标表示。如图 1 中序号为 1、2 的数据点，由于它们处于某一个类别的边界，将会因为其对应的类心密度远远小于阈值 R_5，

而被筛选出来不让其参与聚类，这样做可以有效地避免随机所选择的聚类中心偏离数据集。

为了保证每个个体 $\boldsymbol{V}$ 中至少有 2 个类心参与聚类，所采取的措施是：当个体 $\boldsymbol{V}$ 中仅有 1 个类心参与聚类时，除这个类心外，在个体 $\boldsymbol{V}$ 中再选择一个密度最大的类心参与个体 $\boldsymbol{V}$ 聚类，并将其配对的标签位阈值修改为 0.5~1 之间的一个实数；若个体 $\boldsymbol{V}$ 中没有类心参与聚类时，则选择该个体 $\boldsymbol{V}$ 中前两个密度最大的类心参与聚类，并将它们配对的标签位阈值修改为 0.5~1 之间的一个实数。

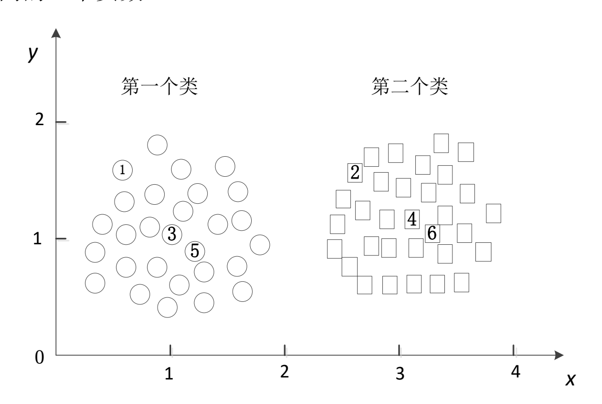


图 1 算法 MODEAC-CD 在模拟数据集 long1 上筛选类心过程

1.3 类心过于集中的改进

以上操作在一定程度上改善了类心偏离数据集的缺陷，但同时也可能使选择的类心过于集中，从而给选择类心聚类带来一定的困难。因此，针对类心过于集中的问题，进一步作了如下改进，具体步骤如下：

a）为了判断两两类心之间是否过于集中，本文采用了文献[11]中定义的一个距离阈值$R_q^{17}=\frac{R_q+R_2}{2}$。其中，$R_q$为 1.2 节中定义的第 q 个类心与其他所有类心之间距离的均值（$q=1,2,\cdots(N)\times K_{max}$）；

b）对于个体 V，找出 V 中参与聚类的且类心密度最大的类心 q_1，求出该类心 q_1 与个体 V 中其余参与聚类的类心之间的欧氏距离，并从中筛选出距离小于阈值$R_{q_1}^{17}$的类心，认为 V 中的这些类心与 V 中类心密度最大的类心 q_1 之间距离过小。为了不让这些类心参与聚类，将其配对的标签位阈值修改为 0~0.5 的一个实数；

c)再从个体 $\boldsymbol{V}$ 中剩下的参与聚类的类心里，找出类心密度第二大的类心 q_2，使用同样的方法筛选出与类心密度第二大的类心q_2距离小于阈值 $R_{q_2}^{17}$的类心（此时，q_1不再参与筛选过程），并将相应类心的标签位阈值修改为 0~0.5 的一个实数，使其不参与聚类；

d)依此类推，直到找出该个体中参与聚类的最后一个类心。

如图 1 所示，序号为 3、4 的数据点是模拟数据集 long1 实际的 2 个类心，而序号为 5、6 这样的数据点，它们是处于某一个真正类别中心周围的类心，将其称为潜在类心。若此时将序号 3 看作是当前个体 $\boldsymbol{V}$ 参与聚类且密度最大的类心，假设序号 5 与序号 3 的距离小于阈值 R_3^{17}，那么按照上面所述，序号 5 这个类心将会被淘汰。以后，若将序号 4 看作是 $\boldsymbol{V}$ 参与聚类且密度第二大的类心，且序号 6 与序号 4 的距离小于阈值 R_4^{17}，那么接下来被淘汰的类心将会是序号 6。这样做可以有效地避免由于所选类心过于集中而带来的错误聚类。

随后，判断个体 $\boldsymbol{V}$ 中参与聚类的类心个数。若小于 2 个（即仅有一个类心密度最大的类心），则求出与类心密度最大的类心距离最远的类心，让其参与聚类，并将其配对的标签位修改为 0.5~1 之间均匀分布的一个随机数。

2 采用类心密度策略的多目标微分自动聚类算法

2.1 聚类准则函数

本文提出的多目标微分自动聚类算法同时优化两种聚类准则函数：误差平方和准则函数$\boldsymbol{G_c}$；改进的类间距离和准则函数$\boldsymbol{G_b}$。

2.1.1 误差平方和准则函数

误差平方和准则函数描述的是类内距离[12],定义如下:

$$\boldsymbol{G_c}=\sum_{j=1}^{c}\sum_{k=1}^{n_j}\left\|\boldsymbol{x_k^{(j)}}-\boldsymbol{C_j}\right\|^2 \tag{2}$$

其中：c 是个体 $\boldsymbol{V}$ 中参与聚类的类心个数，$\boldsymbol{C_j}(j=1,2,\cdots c)$是第 j 类的类心，n_j是分配到第 j 类中的数据点个数，$\boldsymbol{x_k^{(j)}}$ 是第 j 类中的第 k 个数据点，$\boldsymbol{G_c}$是数据点和类心间距离的函数，它描述的是 n 个数据点聚集成 c 个类时，所产生的总的误差平方和。该值越小，表明聚类效果越好。

2.1.2 类间距离和准则函数

类间距离和准则函数描述的是类与类之间的距离[13]，定义如下：

$$\boldsymbol{G_{b1}}=\sum_{j=1}^{c}p_j(\boldsymbol{m_j}-\boldsymbol{m})^T(\boldsymbol{m_j}-\boldsymbol{m}) \tag{3}$$

其中：$\boldsymbol{m_j}$ 为第 j 类中所有数据点每一维的均值构成的向量，$\boldsymbol{m}$ 为全部数据点每一维的均值构成的向量。而 p_j为第 j 类中所有数据点的先验概率，它描述的是第 j 类中数据点个数占所有数据点总数的比率。$\boldsymbol{G_{b1}}$描述了不同类心之间的分离程度，该值越大，表明聚类质量越高。

式（3）中的类间距离和准则函数$\boldsymbol{G_{b1}}$，描述了不同类心之间的分离程度，该值越大，表明聚类质量越高。从公式中可以看出，为了使$\boldsymbol{G_{b1}}$尽可能大，在演化初期，个体更倾向于找出更多的类心。为了使算法能够较快地得到最优的类心个数，从而得到有效的聚类划分，所提算法对原先的 $\boldsymbol{G_{b1}}$ 进行了改进，动态地惩罚类心数目 c。改进的类间距离和准则函数 $\boldsymbol{G_b}$ 描述为

$$\boldsymbol{G_b}=1/c' \times \boldsymbol{G_{b1}} \tag{4}$$

其中：$\boldsymbol{G_{b1}}$ 按式（3）计算所得，$c'=c^{2^k}$，$k=\begin{cases}1-2\times t/t_{max} & t<0.5\times t_{max}\\ 0 & \text{其他}\end{cases}$，$t$ 为当前迭代次数，t_{max}为最大迭代次数。当$t<0.5\times t_{max}$时，c 被动态地惩罚，t 越小，$1/c'$ 越小；而当$t\geq 0.5\times t_{max}$时，对 c 不作惩罚，即$c'=c$。基于改进的类间距离和准则函数 $\boldsymbol{G_b}$，使得算法能够在演化初

期避免找出太多的类心个数，从而影响后面的演化过程，有效地提高了进化效率。

2.2 解的选择策略

多目标聚类算法最终求得的不是一个单一的聚类解，而是一组 Pareto 最优聚类解，这些单独的分组对应着目标之间的不同权衡。而本文求解的聚类问题的结果应为一个具体的最优聚类方案。因此，所提多目标微分自动聚类算法在求得一组 Pareto 最优聚类解后，还需要一个最优解的选择过程。目前常用的解选择方法有 Ding 等人[14]提出的 Gap Statistic，用于评估聚类的个数；Nafchi 等人[15]提出利用 Pareto 最优解集的 MS（Minkowski score）指标值作为选解策略，选择 MS 指标最小的解作为聚类问题的最优解。本文算法从求得一组 Pareto 最优聚类解中，选择准确率[10]最高的解作为最后聚类得到的最优解，该准确率衡量了预测正确的数据量占总预测数据量的比率。

2.3 所提算法 MODEAC-CD 的流程

a)初始化。微分进化算法中的变异算子和交叉算子各包含一个参数：变异因子 F 和交叉因子 CR。设置 F 和 CR 的值（具体取值见 3.3 节参数设置），设置算法的最大目标评价次数为 nmb_obj_max，规定 $nArchive$ 为外部存储器 $\boldsymbol{Archive}$ 最大容量，根据式（1）的个体表示方式，生成规模为 N 的初始父代群体 $\boldsymbol{P}$；

b)对父代群体 $\boldsymbol{P}$ 中每一个个体执行如下操作：

(a)根据 1.2 节、1.3 节筛选类心；

(b)更新类心后对数据集进行聚类，即将每个数据点归类到与其距离最短的类心所在的类中，并依据式（2）（4）计算出个体的目标值，令目标评价次数计数器 $nmb_obj = N$；

(c)确定出$\boldsymbol{P}$中的非支配解集合，并将其存于外部存储器 $\boldsymbol{Archive}$ 中。

c)对父代群体 $\boldsymbol{P}$ 进行微分进化操作，生成子代群体 $\boldsymbol{Q}$，具体生成步骤如下：

(a)首先对群体 $\boldsymbol{P}$ 采用$DE/rand/2$变异策略和二项式交叉策略，生成子代群体 $\boldsymbol{NPOP1}$，$DE/rand/2$ 变异策略和二项式交叉策略的表达式分别如式（5）（6）所示。

$$\boldsymbol{v_i} = \boldsymbol{x_{r1}} + F(\boldsymbol{x_{r2}} - \boldsymbol{x_{r3}} + \boldsymbol{x_{r4}} - \boldsymbol{x_{r5}}) \tag{5}$$

$$u_{i,j} = \begin{cases} v_{i,j}(rand_j[0,1] \le CR)或(j = jrand) \\ x_{i,j} else \end{cases} \tag{6}$$

其中：变异策略 $DE/rand/2$ 中的 3 个随机个体 $\boldsymbol{x_{r1}}$、$\boldsymbol{x_{r3}}$、$\boldsymbol{x_{r5}}$ 取自群体 $\boldsymbol{P}$，另两个 $\boldsymbol{x_{r4}}$、$\boldsymbol{x_{r2}}$ 取自外部存储器 $\boldsymbol{Archive}$，且 $\boldsymbol{x_{r1}} \ne \boldsymbol{x_{r2}} \ne \boldsymbol{x_{r3}} \ne \boldsymbol{x_{r4}} \ne \boldsymbol{x_{r5}}$，变异因子 F 是区间[0.8,0.9]内均匀产生的随机数，$\boldsymbol{v_i}$ 是经过变异后得到的第 i 个中间个体，$rand_j[0,1]$ 为[0,1]之间满足均匀分布的随机数。$jrand$ 为从 $\{1,2,\cdots d\}$ 中均匀随机产生的整数，d 为决策变量的维数，$x_{i,j}$ 为当前个体 $\boldsymbol{x_i}$ 的第 j 维，$v_{i,j}$ 为中间个体 $\boldsymbol{v_i}$ 的第 j 维，$u_{i,j}$ 为子代个体 $\boldsymbol{u_i}$ 的第 j 维；

(b)对$\boldsymbol{P}$再采用$DE/current-to-best/1$ 变异策略和二项式交叉策略，生成子代群体 $\boldsymbol{NPOP2}$。$DE/current-to-best/1$ 变异策略如式（7）所示。

$$\boldsymbol{v_i} = \boldsymbol{x_i} + F_1(\boldsymbol{x_{best}} - \boldsymbol{x_i}) + F_2(\boldsymbol{x_{r2}} - \boldsymbol{x_{r3}}) \tag{7}$$

其中：两个随机个体 $\boldsymbol{x_{r2}}$、$\boldsymbol{x_{r3}}$ 和当前个体 $\boldsymbol{x_i}$ 取自 $\boldsymbol{P}$，$\boldsymbol{x_{best}}$ 个体随机取自外部存储器 $\boldsymbol{Archive}$，且$\boldsymbol{x_{r2}} \ne \boldsymbol{x_{r3}} \ne \boldsymbol{x_i} \ne \boldsymbol{x_{best}}$，变异因子 F_1、F_2 是区间[0.8,0.9]内均匀产生的随机数；

(c)将生成的子代群体合并，得到子代群体 $\boldsymbol{Q}$，即 $\boldsymbol{Q} = \boldsymbol{NPOP1} \cup \boldsymbol{NPOP2}$；

d)根据 b)中的步骤(a)(b)，对生成的子代群体$\boldsymbol{Q}$中的每个个体执行类似操作，累计目标评价次数为$nmb_obj = nmb_obj + |Q|$（|.|表示集合中元素个数）,求出$\boldsymbol{Q}$中的非支配解集，并将其与$\boldsymbol{Archive}$合并；

e)更新与裁减父代群体 $\boldsymbol{P}$ 和外部存储器 $\boldsymbol{Archive}$。令 $\boldsymbol{P} = \boldsymbol{P} \cup \boldsymbol{Q}$，若 $|\boldsymbol{P}| > N$，则采用基于指标 $I_{\varepsilon+}$ 的方式更新 $\boldsymbol{P}$；从 $\boldsymbol{Archive}$ 中确定出非支配解，并赋给 $\boldsymbol{Archive}$，若$|\boldsymbol{Archive}| > nArchive$，则采用基于 Lp-范数距离的多样性维护策略对 $\boldsymbol{Archive}$ 进行更新；对该过程作如下几点补充：

(a) $I_{\varepsilon+}$ 指标描述了在目标空间中，一个解支配另一个解所需要的最小距离[16]，其公式如下：

$$I_{\varepsilon+}(\boldsymbol{x_1},\boldsymbol{x_2}) = min_{\varepsilon}(f_i(\boldsymbol{x_1}) - \varepsilon \le f_i(\boldsymbol{x_2}) 1 \le i \le m) \tag{8}$$

其中:m 为目标维数；根据这个指标为个体分配相应的适应值，式（9）给出了个体 $\boldsymbol{x_1}$ 的适应值计算方法[17]：

$$\boldsymbol{F}(\boldsymbol{x_1}) = \sum_{\boldsymbol{x_2} \epsilon \mathbf{p} \backslash \{\boldsymbol{x_1}\}} -e^{-I_{\varepsilon+}(\boldsymbol{x_2},\boldsymbol{x_1})/0.05} \tag{9}$$

对群体进行更新时，将适应值小的个体依次从群体中移除，直到满足规定的群体规模为止。

(b)Lp-范数的定义如下[18]：

$$Lp(\boldsymbol{x},\boldsymbol{y}) = [\sum_{i=1}^{d}(x_i - y_i)^p]^{1/p} \tag{10}$$

其中:d 表示决策向量的维数，$Lp(\boldsymbol{x},\boldsymbol{y})$ 表示在d维决策空间中，向量 $\boldsymbol{x} =$（$x_1,\cdots,x_d$）与向量 $\boldsymbol{y} =$（$y_1,\cdots,y_d$）之间的 Lp-范数距离。当 $\boldsymbol{Archive}$ 中的个体数量超出规定的容量时，首先将 $\boldsymbol{Archive}$ 中在每个目标上具有最大/最小目标值的个体加入一个空的外部存储器 $\boldsymbol{Archive'}$ 中，然后每次从 $\boldsymbol{Archive}$ 中，选择距离 $\boldsymbol{Archive'}$ 中现有个体最短 Lp-范数距离值最大的个体加入 $\boldsymbol{Archive'}$，如此反复直到 $\boldsymbol{Archive'}$ 中解的个数达到最大容量 $nArchive$。此时，令$\boldsymbol{Archive} = \boldsymbol{Archive'}$。

f)判断终止条件。如果目标评价次数 nmb_obj 达到对应问题的 nmb_obj_max，则算法停止，将当前外部存储器 $\boldsymbol{Archive}$ 作为近似的 Pareto 最优解集，选择其中准确率最高的一个解作为最优解输出，否则，转至 Step3。

2.4 算法时间复杂度分析

设问题的目标个数为 m，群体规模为 N，所提算法 MODEAC-CD 的时间复杂度主要包括以下几个方面：a）在对类心筛选的过程中，其最大时间复杂度为$O((K_{max}N)^2) = O(nN^2)$；b）从生成的子代群体中确定非支配解其复杂度为 $O(Nlog^{m-2}N)$[17]；c）采用基于指标的方式更新群体 $\boldsymbol{P}$，它的复杂度为$O(N^2)$[17]；d）采用支配关系更新 $\boldsymbol{Archive}$，它在比较个体间支配关系时的复杂度为$O(Nlog^{m-2}N)$ [17]，在计算个体间的 Lp-范数距离以维护多样性时的复杂度为$O(mN^2)$ [17]。一般情况下，数据集中数据点的个数$n >$目标个数m,因此，所提算

法在最坏情况下总的时间复杂度取上述分析结果中的最大值 $\max\{O(Nlog^{m-2}N),O(nN^2)\}$。

3　仿真实验

3.1　实验数据描述

为了验证所提算法 MODEAC-CD 在数据聚类上的效果，选择了两组不同类型的实验数据。第一组数据集是 4 个 UCI(University of California, Irvine)数据集，该数据集是一种在数据挖掘中常用的公共标准测试集[19]。第二组数据集是 4 个具有球形数据特征的人工数据集[10]。具体的数据属性如表 1 所示。

表 1　数据属性描述

数据集	数据数目（n）	数据维数（d）	类心数（c）
UCI 数据集			
Diabetes	768	8	2
Iris	150	4	3
Glass	214	9	6
Liver	345	6	2
人工数据集			
Square1	1000	2	4
Square4	1000	2	4
Long1	1000	2	2
AD_5_2	250	2	5

3.2　比较算法介绍

文献[10]设计了两种基于微分进化的自动聚类算法 PSIMACDE 和 DEAFC_DO，并将这两种算法与经典的三种自动聚类算法 GADE[20]、VGAPS[21]、ACDE[4]在一组数据集上进行了比较。结果显示，算法 PSIMACDE 和 DEAFC_DO 无论是在聚类的类别数还是准确率方面，在绝大多数数据集中均优于其余几种经典的自动聚类算法。2014 年，Rodriguez 等人在《Science》上提出了密度峰聚类算法（RLCLu）[7]。该算法介绍了一个数据点被看作是聚类中心，需要满足两个基本条件：该数据点被密度相对较低的邻域点所包围；该点与其他密度更高的点之间距离应相对较大。算法 RLCLu 通过对合成数据点分布以及人脸数据库数据的测试，证明了该算法能够识别出具有不同形状和维数的数据集类心。

为了检验本章所提算法 MODEAC-CD 的有效性，本章将它与上述算法 PSIMACDE、DEAFC_DO 以及算法 RLCLu 进行了比较。其中，前三种算法均是基于微分进化的自动聚类，它们的区别在于：所提算法 MODEAC-CD 采用类心密度策略筛选聚类中心，使用指标$\boldsymbol{G_c}$和$\boldsymbol{G_b}$作为聚类准则函数，提高了聚类的准确性；算法 PSIMACDE 使用免疫克隆的思想来保持群体的多样性，并使用指标 *XB*[22]和 *Sym_index*[23] 作为聚类准则函数；算法 DEAFC_DO 采用单个指标 *PMB*[24]作为聚类准则函数，并提出了一种基于类别中心密度排序的类心数振荡策略，提高了算法的局部搜索能力。所提算法 MODEAC-CD 和算法 RLCLu 的主要区别是：所提算法能够自动识别聚类中心，而算法 RLCLu 没有使用任何指标作为聚类准则函数且只能通过决策图人工选取类心，从而增加了类心选取的不确定性。

3.3　参数设置

在每个聚类问题中，将前三种算法各自独立运行 20 次以统计其结果。为了保证算法对比的公平性，规定前 3 种算法的停止准则都为目标评价次数达到给定的最大值即 $nmb_obj_max=30020$。所提算法的参数设置依据实验结果调试得到，算法 PSIMACDE 和算法 DEAFC_DO 的参数设置参照文献[10]，各算法具体取值如表 2 所示。前三种算法的交叉概率 *CR* 在 $[CR_{min},CR_{max}]$ 中随机生成。对于第 4 种算法 RLCLu 而言，在不改变输入参数的情况下，其每次运行所得的聚类决策图结果均一样，所以操作者可以通过决策图人工选取局部密度相对较高以及与其它密度更高的点之间距离相对较大的点作为类心，来统计其聚类结果。其缺陷是对于一些特殊数据集，通过决策图人工选取聚类中心时容易出错。该算法只需提前定义每个点的邻域个数占总个数的百分比即可，这里定义百分比参数 percent=2 。

表 2 参数设置

算法	MODEAC-CD	PSIMACDE	DEAFC-DO
F	$0.8+\mathrm{rand}[0,1]\cdot 0.1$	$0.5\cdot(\mathrm{rand}[0,1]+1)$	$0.5\cdot(\mathrm{rand}[0,1]+1)$
Popsize	20	20	20
K_{max}	20	20	20
CR_{max}	1.0	1.0	1.0
CR_{min}	0.5	0.5	0.5
t_{max}	500	500	500

3.4　实验结果与分析

本文使用三种测度：聚类类别数、聚类准确率[10]、adjusted rand index(ARI)[25]以及作图法将所提算法 MODEAC-CD 与已有算法 PSIMACDE、DEAFC_DO 以及算法 RLCLu 进行比较。其中，算法所得聚类类别数，越接近数据真实的聚类数越好；算法所得聚类正确率越高越好；对于评价标准 ARI 来说，有两个输入变量，一个是正确的划分结果，另一个是实验所得的划分结果，它统计了所有数据项在这两种划分结果中成对出现在同一类中的机率，该值越大聚类效果越好。

1）作图法实验结果与分析

图 2、3 是四种算法分别独立运行 20 次后，在数据集 AD_5_2 和数据集 square4 上，得到的最好聚类结果。其中 *x* 轴，*y* 轴为各个数据点在 2 维空间上的坐标表示。从图 2 可以看出，算法 MODEAC-CD、PSIMACDE 以及算法 RLCLu 将数据集 AD_5_2 分成了 5 类，与该数据集实际分类数目相符；而算法 DEAFC_DO 只将该数据集分成了 4 类，不符合实际分类数目。从图 3 中可以看出，算法 MODEAC-CD、PSIMACDE 以及算

法 RLCLu 将数据集 square4 分成了 4 类，与该数据集实际分类数目相符；此外，可观察到，所提算法 MODEAC-CD 得到的数据分布比算法 PSIMACDE 以及算法 RLCLu 更为均匀合理，这得益于所提算法提出的类心密度策略。而算法 DEAFC_DO 仅将数据集 square4 分成了 2 类，不符合该数据集的实际分类数目。出现这种情况，主要因为算法 DEAFC_DO 为单目标自动聚类算法，而其余两种基于微分进化的算法为多目标自动聚类算法。因此，多目标聚类算法在整体上可以获得比单目标聚类算法更优的划分结果。这得益于多目标聚类方法采用了两个目标函数，同时优化了聚类的类内距离和类间距离两个指标，因此可以提高解的聚类质量，避免只偏向于一个目标函数的缺点。同时，也可以看出和其它 3 种算法相比，算法 DEAFC_DO 不善于处理那些分布较为紧凑且相对复杂的数据集。此外，由于算法 RLCLu 在聚类过程中，同时考虑了类心的局部密度以及类心之间的距离，所以也取得了不错的聚类效果。从图 3 的(*d*)中也可以看到，在这些由各个类核心点组成的类的周围还分布着许多其它的点，我们将这些点称作噪声点或者离群点，而算法 RLCLu 很好地将这些我们不需要的点筛选了出来，表明了该算法具有不错的聚类性能。

2）聚类数及 ARI 测度的实验结果与分析

将四种算法在两种类型的数据集上独立运行 20 次后，表 3 给出了各算法得到的聚类数及 ARI 的均值和方差。

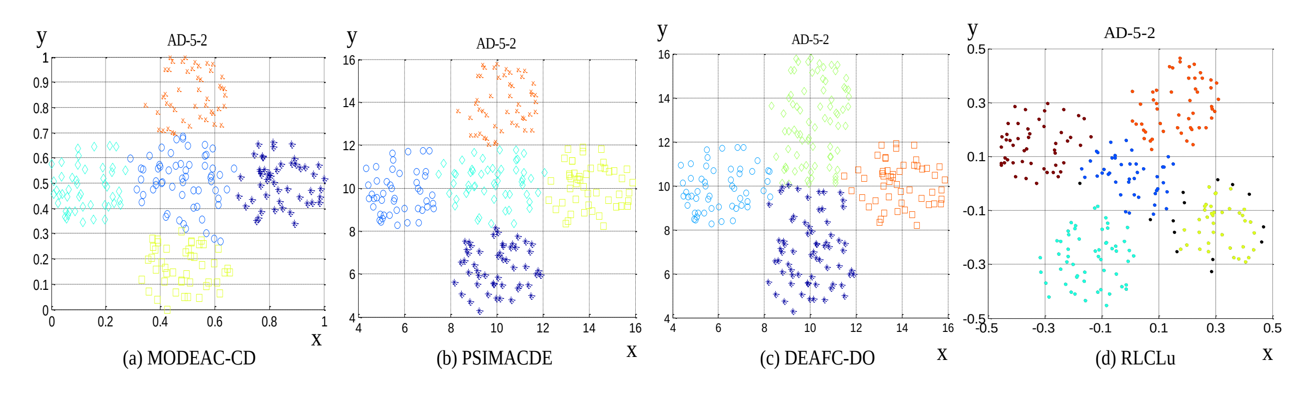


图 2　4 种算法在数据集 AD_5_2 上的聚类结果

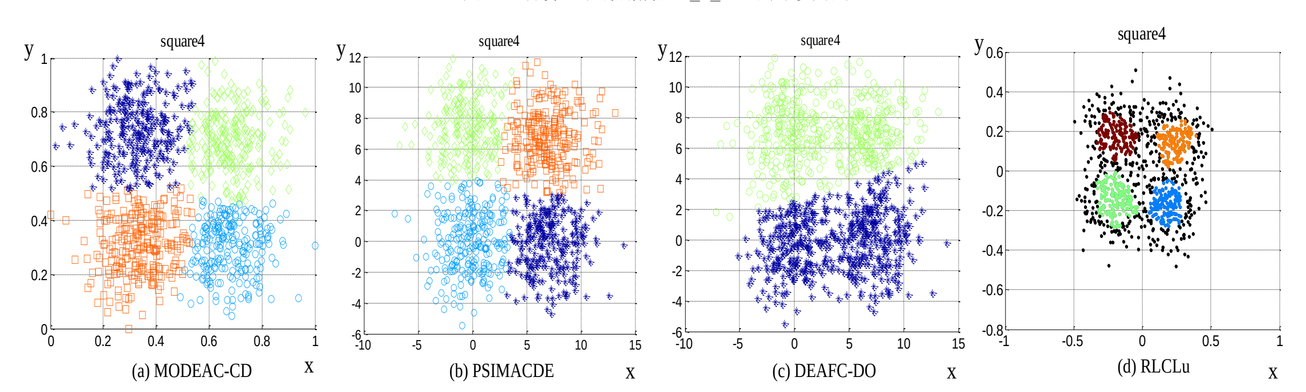


图 3　4 种算法在数据集 Square4 上的聚类结果

表 3　聚类数及测度 ARI 的均值和方差

		MODEAC-CD		PSIMACDE		DEAFC_DO		RLCLu	
UCI 数据集	实际类心数	聚类数	ARI	聚类数	ARI	聚类数	ARI	聚类数	ARI
Diabetes	2	**2.00±0.000**	**0.3759±0.0343**	2.30±0.732	0.3617±0.0137	2±0	0.3363±0.0278	2.40±0.547	0.2595±0.0072
Iris	3	**3.05±0.223**	0.8244±0.0591	4.25±2.099	**0.8547±0.0606**	2±0	0.8162±0.0066	2.75±2.750	0.6370±0.2274
Glass	6	**4.90±1.209**	**0.4608±0.0307**	4.10±1.165	0.4178±0.0278	2±0	0.4221±0.0038	2.75±0.957	0.3220±0.0723
Liver	2	**2.00±0.000**	**0.2520±0.0050**	2.50±0.827	0.2483±0.0050	2±0	0.2454±0.0010	3.80±1.169	0.2461±0.0016
人工数据集	实际类心数	聚类数	ARI	聚类数	ARI	聚类数	ARI	聚类数	ARI
Square1	4	4.05±0.223	0.9450±0.0206	4.05±0.223	0.9650±0.0189	**4±0**	**0.9696±0.0030**	3.50±0.707	0.8405±0.1954
Square4	4	**4.00±0.000**	0.8014±0.0380	4.80±1.056	**0.8333±0.0500**	2±0	0.6634±0.0142	3.00±1.000	0.7748±0.0855
Long1	2	**2.00±0.000**	**1.0000±0.0000**	2.00±0.000	1.0000±0.0000	2±0	1.0000±0.0000	2.00±0.000	1.0000±0.0000
AD_5_2	5	**5.00±0.324**	**0.8788±0.0473**	5.35±0.875	0.8536±0.0622	4±0	0.8100±0.1189	4.80±1.483	0.7776±0.0910

从表 3 中各算法得到的聚类数均值和方差情况，我们可以看出，所提算法 MODEAC-CD 在测试的 8 组数据集中，有 7 组数据集的聚类数最好，也最接近数据集的实际类数，并且所得聚类方差也最小，从而表明了所提聚类算法具有很好的稳定性。算法 PSIMACDE 在人工数据集上得到的聚类数总体表现效果不错。算法 DEAFC_DO 在人工数据集 Square1、Long1 上，UCI 数据集 Diabetes、Liver 上得到的聚类数表现不错。算法 RLCLu 在数据集 Iris、Long1、AD_5_2 上表现出不错的聚类性能。

从表中各算法得到的 ARI 测度均值和方差情况，我们可以看出：与其余三种算法相比，所提算法 MODEAC-CD 在 UCI 数据集四组数据中的三个表现出了最好的聚类效果，在人工数据集 long1 和 AD_5_2 上取得了最好的聚类性能。算法 PSIMACDE 在数据集 Iris、Square4 上表现出最好的聚类性能。算法 DEAFC_DO 在数据集 Square1 上表现出最好的聚类性能。算法 RLCLu 在人工数据集聚类方面表现出相对不错的聚类性能。此外，可以观察到这四类算法在 UCI 数据集上得到的 ARI 值总体效果并不是很好，这也验证了随着数据集特征维数的增多，数据正确聚类的难度也随之增加。

表 4 给出了各算法独立运行 20 次后，得到的聚类准确率均值和方差。

表 4　准确率的均值和方差

	MODEAC-CD	PSIMACDE	DEAFC_DO	RLCLu
UCI 数据集准确率的均值和方差				
Diabetes	**89.2643±1.8566**	84.9740±5.5603	86.7253±1.0616	83.5677±2.1463
Iris	89.4667±4.2473	**91.8000±3.4069**	86.6667±0.0000	81.833±11.4746
Glass	**69.6495±3.1977**	67.5700±1.4668	63.4813±0.1045	60.8879±5.4025
Liver	**77.0870±0.7839**	74.3478±5.1736	77.0580±0.2160	70.7246±7.7213
人工数据集准确率的均值和方差				
Square1	97.8750±0.8213	98.4000±1.9404	**98.8550±0.1150**	96.9000±3.2527
Square4	**91.7650±0.6459**	91.4950±1.9014	68.9050±0.5316	83.933±13.1702
Long1	**100.000±0.0000**	100.000±0.0000	100.000±0.0000	100.000±0.0000
AD_5_2	**94.2600±2.2338**	94.0600±2.4701	79.5800±0.6678	84.8000±7.1889

表 4 结果显示，所提算法 MODEAC-CD 在六组数据集上的聚类准确率最高，算法 PSIMACDE 在数据集 Iris 上的准确率最高，算法 DEAFC_DO 在数据集 Square1 上的聚类准确率最高，而算法 RLCLu 在数据集 Square1 和 Long1 上的准确率相对不错。此外，可以看出这四种算法在数据集 Long1 上的聚类准确率均为 100% ，表明了各算法均对该数据集进行了正确的聚类。

表 5 给出了各算法独立运行 20 次后，得到的运行时间均值。

表 5　算法运行时间的比较

	MODEAC-CD	PSIMACDE	DEAFC_DO	RLCLu
UCI 数据集运行时间的均值（s）				
Diabetes	130.942	29.317	36.353	4.861
Iris	109.353	5.780	8.275	4.557
Glass	156.049	7.222	11.183	4.980
Liver	129.090	11.311	17.552	3.884
人工数据集运行时间的均值（s）				
Square1	93.189	36.860	62.644	7.405
Square4	88.912	36.680	44.970	6.803
Long1	81.773	36.084	41.703	4.482
AD_5_2	72.550	9.718	14.289	6.796

由表 5 可见，与其他三种算法相比，所提算法 MODEAC-CD 花费的平均运行时间相对较长。原因是所提算法采用的类心筛选策略需要消耗比较多的计算资源，但该策略能够有效地选择较好的聚类中心，改善了由于算法随机性导致的类心容易偏离数据集的缺陷，同时防止选择的类心过于集中。考虑到本文研究的数据自动聚类问题是一类离线优化问题，对算法的实时性要求不高，因此所提算法的运行时间在可接受的范围内。

综合表 3 和 4 可以发现，所提算法 MODEAC-CD 在大部分数据集上得到的类别数、聚类正确率以及 ARI 指标值均最好，体现了所提算法策略设计的有效性。此外，也可以看出：

a）所提算法 MODEAC-CD 在大部分数据集上，得到的方差均较小，表明了所提算法具有较稳定的搜索性能。

b）在大多数问题中，如果算法所得聚类类别数越接近数据集真实类别数，则其聚类准确率越高，测度 ARI 值也越大。

c）这四种算法在解决具有球形数据特征的人工数据集时，总体上能够表现出更好的聚类效果，而相比之下，在解决数据特征维数较高或较为复杂的数据集（如 UCI 数据集）时，还表现出不足，有待进一步研究。

d）各算法在数据集 square4 和数据集 AD_5_2 所得结果，很好地验证了图 2、3 所呈现的效果。

4 结束语

为了在事先不知道待聚类问题聚类个数的情况下，仍然能够较为准确地聚类，本文采用了实数定长的个体编码方式以实现自动聚类；为了有效避免因算法本身的随机性导致的错误聚类，本文提出了采用类心密度策略的多目标微分自动聚类算法 MODEAC-CD。所提类心密度策略能够在数据集聚类前，对算法随机选择的某些偏离数据集或者分布过于集中的类心进行筛选，不让其参与聚类；为了使算法能较快地得到最优类心，提出了改进的聚类准则函数，对聚类数目进行动态地惩罚。将所提算法 MODEAC-CD 和其余两种性能较优的自动聚类算法以及经典聚类算法 RLCLu，在两种不同类型的数据集上进行了比较，结果表明，所提算法 MODEAC-CD 具有更优的聚类效果，它在大多数问题中得到的聚类数目与数据集实际分类数目相符或更为接近，且具有更高的 ARI 性能和聚类准确率，从而表明本文算法采用的策略是可行而有效的。

然而，在聚类的过程中，发现所提算法对某些复杂结构数据集的聚类效果不是很好。如本文中提到的 glass 数据集，该数据集中各簇类所包含的数据个数差异悬殊，并且各数据之间分布相对紧密，使得算法对其聚类造成了一定困难。因此，如何选择更有效的聚类机制，解决此类数据集聚类问题，将是下一步研究的内容。

参考文献：

[1] Zitler E, Deb K, Thiele L. Comparison of multi-objective evolutionary algorithms: empirical results [J]. Evolutionary Computation, 2000, 8 (2): 173-195.

[2] Storn R, Price K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces [J]. Journal Global Optimization, 1997, 11 (4): 341-359.

[3] Tam H H, Ng S C, Andrew L, *et al.* Improved activation schema on automatic clustering using differential evolution algorithm [C]// Proc of IEEE Congress on Evolutionary Computation. 2017: 1749–1756.

[4] Swagatam D, Ajith A, Amit K. Automatic clustering using an improved differential evolution algorithm [J]. IEEE Trans on Systems, 2008, 38 (1): 218-237.

[5] Saha I, Maulik U, Bandyopaghyay S. A new differential evolution based fuzzy clustering for automatic cluster evolution [C]// Proc of IEEE International Advance Computing Conference. 2009: 706-711.

[6] Maulik U, Saha I. Automatic fuzzy clustering using modified differential evolution for image classification [J]. IEEE Trans on Geoscience and Remote Sensing, 2010, 48 (9): 3503-3510.

[7] Rodriguez A, Laio A. Clustering by fast search and find of density peaks [J]. Science, 2014, 344 (6191): 1492-1496.

[8] 李涛，葛洪伟，苏树智. 自动确定聚类中心的密度峰聚类 [J]. 计算机科学与探索, 2016, 11 (10): 1614-1622. (Li Tao, Ge Hongwei, Su Shuzhi. Automatic determination of density peak clustering in cluster center [J]. Computer Science and Exploration, 2016, 11 (10): 1614-1622.)

[9] Ye Xuanzuo, Li Dinghao, He Xiongxiong. An algorithm for automatic recognition of cluster centers based on local density clustering [C]// Proc of the 29th Chinese Control and Decision Conference. 2017: 1347-1351.

[10] 武小龙. 基于改进的差分进化自动聚类算法研究 [D]. 西安: 西安电子科技大学, 2013. (Wu Xiaolong. Research on improved automatic clustering algorithm based on differential evolution [D]. Xi’an: Xidian University, 2013.)

[11] 李建. 聚类融合研究及其应用 [D]. 哈尔滨: 哈尔滨工程大学, 2014. (Li Jian. Research and application of cluster fusion [D]. Harbin: Harbin Engineering University, 2014.)

[12] 张素洁，赵怀慈. 最优聚类个数和初始聚类中心点选取算法研究 [J]. 计算机应用研究, 2017, 34 (6): 1617-1620. (Zhang Sujie, Zhao Huaici. Research on optimal clustering number and initial clustering center selection algorithm [J]. Application Research of Computers, 2017, 34 (6): 1617-1620.)

[13] 黎凡，王新，和晓萍. 一种基于局部密度的 K-means 算法 [J]. 云南民族大学学报：自然科学版, 2014, 23 (6): 439-442. (Li Fan, Wang Xin, He Xiaoping. A K-means algorithm based on local density [J]. Journal of Yunnan famous university: Natural Science Edition, 2014, 23 (6): 439-442.)

[14] Ding Jie, Noshad M, Tarokh V. Learning the number of autoregressive mixtures in time series using the gap statistics [C]// Proc of the 15th IEEE International Conference on Data Mining. 2015: 1441-1446.

[15] Nafchi H Z, Shahkolaei A. Mean deviation similarity index: efficient and reliable full-reference image quality evaluator [J]. IEEE Access. 2016 (4): 5579 – 5590.

[16] Zitzler E, Kunzl S. Indicator-based selection in multi-objective search, ”in parallel problem solving from nature [C]// Proc of International Conference on Parallel Problem Solving from Nature. 2004: 832–842.

[17] Wang Handing, Jiao Licheng, Yao Xin. Two_Arch2: an improved two-archive algorithm for many-objective optimization [J]. IEEE Trans on Evolutionary Computation, 2015, 19 (4): 524-541.

[18] Aggarwal C C, Hinneburg A, Keim D A. On the surprising behavior of

distance metrics in high dimensional space [C]// Proc of International Conference on Database Theory. 2001: 420-434.

[19] Zhang Li, Zhang Chengjin, Xu Qingyang, *et al.* Weigted-KNN and its application on UCI [C]// Proc of IEEE International Conference on Information and Automation. 2015: 1748-1750.

[20] KunduD, SureshK, GhoshS. Automatic clustering using a synergy of g enetic algorithm and multi-objective differential evolution [C]// Proc of International Conference on Hybrid Artificial Intelligence Systems. 2009: 177-186.

[21] Bandyopadhyay S, Saha S. A point symmetry based clustering technique for automatic evolution of clusters [J]. IEEE Trans on Knowledge and Data Engineering, 2008, 20 (11): 1–17.

[22] Gao Bo, Wang Jun. Multi-objective fuzzy clustering for synthetic aperture radar imagery [J]. IEEE Geoscience and Remote Sensing Letters, 2015, 12 (11): 2341-2345.

[23] 龚文引. 差分演化算法的改进及其在聚类分析中的应用研究 [D]. 武汉：中国地质大学, 2010. (Gong Wenyin. Improvement of differential evolution algorithm and its application in clustering analysis [D]. Wuhan: China University of Geosciences, 2010.)

[24] Ashok P, Kadhar G M. Detecting outliers on uci repository datasets by adaptive rough fuzzy clustering method [C]// Proc of Online International Conference on Green Engineering and Technologies. 2016: 1-6.

[25] Park S, Choi H, Lee B. hc-OTU: a fast and accurate method for clustering operational taxonomic units based on homopolymer compaction [J]. IEEE//ACM Trans on Computational Biology and Bioinformatics, 2018, 15 (2): 441-451.